Virtage: Server Virtualization with Hardware Transparency

Hitoshi Ueno, Satomi Hasegawa, and Tomohide Hasegawa

Enterprise Server Division, Hitachi, Ltd.
Hadano, Japan
{hitoshi.ueno.uv,satomi.hasegawa.ch,
tomohide.hasegawa.tv}@hitachi.com

Abstract. This paper presents Virtage, Hitachi's virtualization technology, which enables logical partitioning of server platforms. Logical partitioning architecture brings two benefits to this virtualization technology, one is hardware transparency and another is better performance.

We first describe its key feature: hardware transparency. With the advantage of hardware transparency for logical servers, it is possible to provide the same guest Operating System (OS) interface from both physical servers (non-virtualized servers) and logical servers. Virtage is hypervisor-type virtualization, and therefore has a natural performance advantage over host-emulation virtualization offerings because guest OSs can be simply and directly executed on the virtualized environment without host intervention.

Then we demonstrate this lower overhead for CPU and I/O with performance experiments and explain how Virtage brings mainframe-class virtualization to blade servers. In this paper we study the factors of virtualization overhead for CPU and I/O by using original event monitoring tool that can get hypervisor-event information.

Keywords: virtualization technology; Virtage; server; pass-through method; hardware transparency; benchmark; scalability.

1 Introduction

Recent developments in server hardware technology have resulted in increased numbers of CPU cores and the need to install greater memory capacity on each server, thereby making it harder to efficiently utilize the full performance of a server with a single Operating System (OS). This problem can be resolved through server virtualization technology, which allows one server to run multiple applications on top of independent OSs. Currently, server virtualization technology is mainly used for server migration from old and low performance systems to new high performance hardware. It is important to have high performance, high availability and easy system management for server consolidation.

Virtage is the new server virtualization technology available on Hitachi's BladeSymphony servers. Virtage implements a logical partitioning system, which presents the physical server hardware configuration to logical servers without abstraction of

H.X. Lin et al. (Eds): Euro-Par 2009 Workshops, LNCS 6043, pp. 404–413, 2010.

the hardware configuration. This is a unique virtualization feature. By using this logical partitioning system, Virtage users can utilize high availability cluster systems or database systems.

The purpose of this paper is to introduce Virtage and this new approach of server virtualization technology. It is suitable for a wide range of business and research applications. We discuss the design aspects of Virtage and its advantages. In the rest of the paper, we describe the key features of Virtage with some examples of system implementation and operations management. We also present some experimental results of Virtage performance.

2 Design Policy

For blade servers—which are often employed in businesses' mission-critical systems—running a virtualized environment, it is desirable to maintain support for high-availability and high-operability systems in order to secure the same high reliability and operational efficiency as physical servers.

Research was carried out in the field of mainframe computing systems in the 1970s in pursuit of higher performance in virtual machines [2], [3]. Even today, products based on logical partitioning are heavily used in mainframe computing systems.

Our development goal is a server product intended for mission-critical systems, which requires a virtualization system suitable for this kind of usage profile. For this reason, Virtage adopts the logical partitioning method.

Virtage has been designed to provide "hardware transparency" for logical servers. It provides the same guest OS interface for both logical servers and physical servers. The logical partitioning of systems provides the following benefits:

- The same file system can be used in both the virtual and the physical server environment. (e.g. NTFS, EXT3, etc.) With this Virtage feature, a system disk which was deployed in a virtual environment can also be used in a physical environment.
- Both a CPU dedicated mode and a CPU shared mode are available. The CPU service ratio can be assigned in units of one percent when using CPU shared mode.
- Less performance overhead caused by the virtualization control.
- In a logical server environment, I/O intensive application software works normally, just as it works on a physical server. Examples are storage management software and cluster control software of hot-standby systems.

In the logical server environment of Virtage, any application software can work normally, because a guest OS and its applications can access I/O devices in the same way they would access the physical server's hardware interface.

2.1 Hardware Transparency

In this section, we discuss hardware transparency, which is a distinctive characteristic of Virtage. Here, hardware transparency is defined according to the two conditions specified below. We will examine what benefit is delivered by hardware transparency when these two conditions are satisfied.

1. All I/O commands issued by the guest OSs (OSs on virtual computers) and their responses must be identical to those on a physical server.
2. The format of the disk used by the guest OSs must be identical to that of a physical server.

If the process executed by a guest OS on the disk is just a simple read/write, condition 1 does not necessarily need to be satisfied. In general, neither condition 1 nor condition 2 is satisfied with other virtualization software. However, in order to return correct responses to the execution of control-related commands issued by cluster control software etc., condition 1 should be satisfied. To put it another way, when this condition is met, a cluster of linked logical servers can be structured without special restrictions.

Meanwhile, two advantages are obtained when condition 2 is met. One advantage is that, because the file system can be accessed directly, the system can be accessed from a backup server that does not support virtualization. This makes it possible to adopt the so-called LAN-free backup configuration, in which the incremental change data of the disk used by the guest OSs is backed up, file by file, via SAN, without going over the LAN.

The other advantage arises from the ability to write data from the guest OSs to the disk unit securely. In general, virtualization software often adopts a virtualized file system as the disk unit's recording format. In this case, it is common to prepare a disk cache in the server's memory. If the virtualization software has a disk cache, data is likely to be retained in the cache for a certain period of time after a data-write-completion-response has been returned, even when the guest OSs execute a raw write (intending to record data securely on a physical disk). Such caching means that the system cannot maintain the journal files necessary to preserve the reliability of transaction monitors and database software. Some virtualization software does have raw write capability: a guest OS can write data directly into disk units, bypassing the virtualized file system. However, to use this capability, users have to consider which volumes must be configured with virtualized file systems and which volumes must be configured with normal file systems. This adds to the complexity of configuring production systems.

2.2 Comparison of I/O Virtualization Methods

There are two typical I/O virtualization methods. One is the pass-through method and the other is the hypervisor emulation method, which is often used in other virtualization software. The pass-through method is desirable in order to achieve hardware transparency.

- Pass-through method

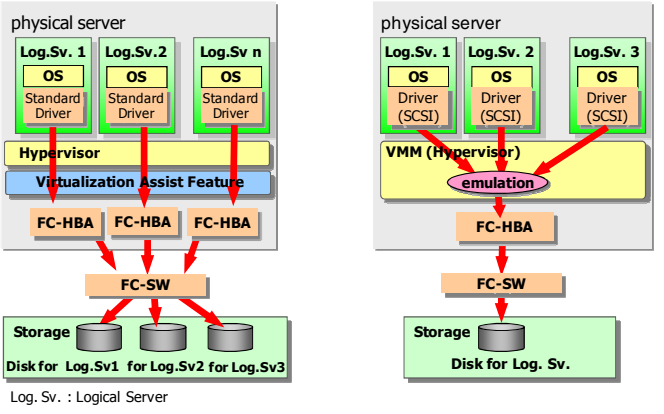
  The pass-through method is a technique that does not need the intervention of the hypervisor when a guest OS activates and executes an I/O operation.

  The DMA address set up by the guest OS on the I/O device is a virtualized memory address. Therefore, if the I/O device executes data transfer as is, the data will be transferred to the wrong memory location. To prevent this error without the intervention of the hypervisor, hardware assistance is required.

- Hypervisor emulation method

  The hypervisor emulation method is a virtualization technique that executes address translation for DMA transfer by guest OSs by means of emulation, using hypervisor traps at the time of I/O activation.

Fig. 1 shows conceptual diagrams of the two virtualization methods, and Table 1 compares their functional features.



(a) Pass-through method (b) Hypervisor emulation method

Fig. 1. Structure of I/O virtualization methods

Table 1. Comparison of I/O virtualization methods

	Pass-through Method		Hypervisor Emulation Method	
I/O performance	G	High performance by direct execution	N	Large overhead due to emulation
Hardware transparency	G	Clustering for hot standby system is available	N	Hard to gain right response for control accesses
File system transparency	G	LAN free backup/ DBMS use available	N	Virtual file system prevents common system use
Disk virtualization	N	not supported	G	Virtual Machine migration available

G: Good, N: Not good

On the basis of the comparison in Table 1, we decided to adopt the pass-through method for virtualization on the BladeSymphony BS1000 servers, which is aimed at businesses' mission-critical systems.

As discussed above, hardware assistance is needed to implement pass-through I/O. We developed a unique I/O virtualization assist mechanism to implement this hardware assistance. Thus, we have succeeded in creating a practical pass-through method.

3 Performance Experiments

In a real-life virtualized environment, some virtualization control overhead cannot be ignored, such as emulation processing. We have made a performance experiments with Virtage. It is a system-level test with a 3-tier application program running on several logical servers (LPARs) on a physical server in order to investigate the scalability of this workload.

In this test, we used a typical ERP workload model, Sell from Stock load Scenario, for the system test because it is CPU and memory intensive and uses substantial amounts of network traffic and disk I/O.

Fig. 2 shows the test environment used. The system used for the experiments consisted of application servers on logical servers (one application server per LPAR), a database server on a separate physical server, and external test drivers connected to the application servers.

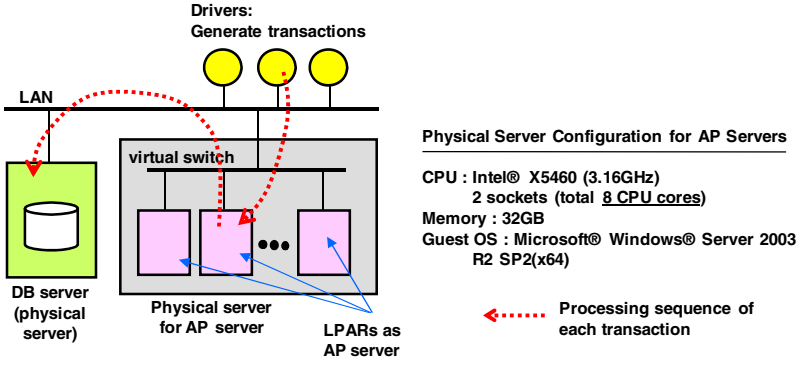


Fig. 2. 3-tier (database server, application servers, and the drivers) application program model and the environment

We evaluated three levels of transaction workloads; heavy, middle, and light with 60%, 40%, and 20% of CPU utilization in the system respectively. The performance is given by measuring the response time of the transactions with these workloads. The test server has eight physical CPU cores. In the test, we used four, six, and eight logical servers (LPARs) as application servers, and each application server had two virtual CPUs. Therefore, a maximum of 16 virtual CPUs is running in this test.

In the case of four running application servers, the number of virtual CPU cores equals the number of physical CPU cores. However, with six and eight application servers, the number of virtual CPU cores exceeds the number of physical CPU cores. We call this processor over-commitment. This ERP workload test is designed to be completed with a two second response time for practical use.

Fig. 3 shows that this condition can be satisfied in the cases of four and six logical servers (LPARs) running simultaneously in one physical server for all levels of transactions. On the other hand, in the test with eight logical servers, this condition is satisfied only for the middle and light transaction workloads.

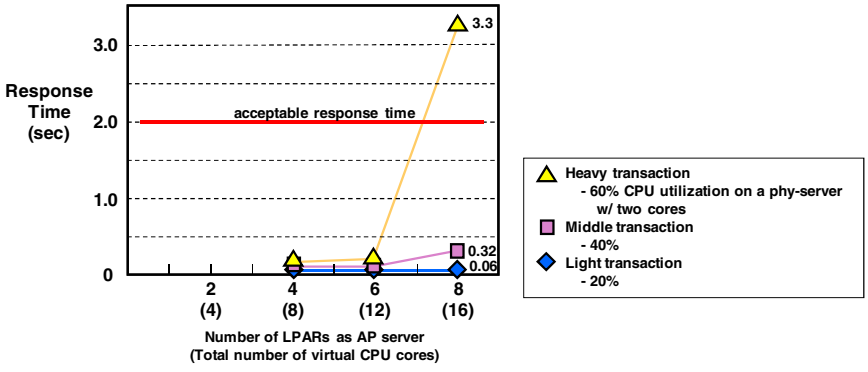


Fig. 3. Virtage performance of ERP workload

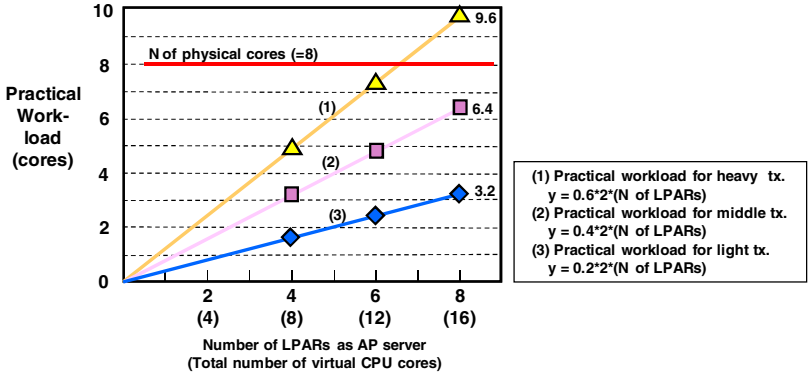


Fig. 4. Practical workload of each transaction case

We can easily understand above phenomenon if we have the graph which plots practical consumed physical CPU cores. Fig. 4 shows numbers of CPU cores that consumed in each case, heavy, middle and light. It shows that only one case exceeds numbers of existing "physical CPU cores" that is 8 and Fig. 3 shows that only this case exceeds the response time limitation. In other words, if numbers of consumed CPU core is less than numbers of existing physical CPU cores, the system achieves enough quick response time.

Therefore, relatively good performance with processor over-commitment is shown for light and middle transaction workloads. Only the heavy transaction workload with eight logical servers cannot achieve the desired performance. We are investigating whether this is due to a performance limitation of this test model (that is, an application limitation) or whether it is due to virtualization overhead.

4 Performance Overhead Analisys

"CPU performance overhead" in server virtualization is the additional processing time which does not exist on physical server processing. From viewpoints of instruction

execution time, some hypervisor instruction streams are inserted in the application and OS instruction stream in virtualization environment. The performance overhead is the additional time to execute the inserted hypervisor instruction streams.

Examples of causes of the inserted hypervisor instruction streams are as follows.

- Critical resource accessing : If guest OSs access critical resources like control registers or I/O registers for MMIO operations, the hypervisor has to emulate the resources.
- Page fault occurrence : If page fault is occurred on guest OSs, the hypervisor has to maintain memory management tables like page table or TLB.
- Interrupt from hardware : If external interruptions are occurred on the hardware, hypervisor has to emulate another interruption for specified guest OS.

It is useful to measure the frequency and execution time to detect the cause of performance overhead and think out the measures, because the frequency of those hypervisor events depends on the characteristic of the executions of application programs.

We developed the event counter and the event tracer functions as a performance monitor and put them into the hypervisor. The event counter counts the hypervisor events and sum up their execution time respectively. The event tracer records the hypervisor events in sequence.

We measured the virtualization overhead factors using Virtage performance monitor in former ERP workload. The Intel® X5460 platform, Virtage uses SPT(Shadow Page Tables) method for guest to host address translation. The overhead is relatively high because the hypervisor execution is needed for address translation. Concretely hypervisor set the guest to host translation to SPT at Page fault and invalidate it at CR access which is issue by OS at guest process switch.

The Intel® X5570 platform, Virtage uses EPT(Extended Page Tables) method for guest to host address translation. In EPT method the address translation is executed by hardware and hypervisor execution is not needed. The overhead is 76.6% less than the one in SPT method.

Table 2. Result of performance monitoring

Factor of VMexit	Grouping	SPT method	EPT method
PF_REASON_INST_EMULATION	MMIO	0.053	0.000
mmioCacheEmulation	MMIO	0.000	0.000
External interrupt	Interrupt	0.033	0.054
Interrupt window	Interrupt	0.003	0.001
CPUID	Instruction Emulation	0.000	0.000
MOV DR	Instruction Emulation	0.000	0.000
I/O instruction	ACPI timer	0.002	0.002
RDMSR	Instruction Emulation	0.000	0.000
WRMSR	Instruction Emulation	0.000	0.000
TPR below threshold	Interrupt	0.003	0.003
APIC(EOI)	Interrupt	0.017	0.033
APIC(other)	Interrupt	0.090	0.088
vmexitEptViol_COST	MMIO	0.000	0.052
HLT(host)	Halt	0.009	0.001
PF_REASON_SPT_UPDATE	Page Fault	0.482	0.000
CR access	CR access	0.306	0.000
INVLPG	MMU	0.001	0.000
PF_REASON_GUEST_PF	MMU	0.001	0.000
Total		1.0	0.234

Table 2 shows the result of the performance monitoring for ERP workload benchmark, and Fig. 5 shows performance overhead analysis results between SPT method and EPT method in the workload case.

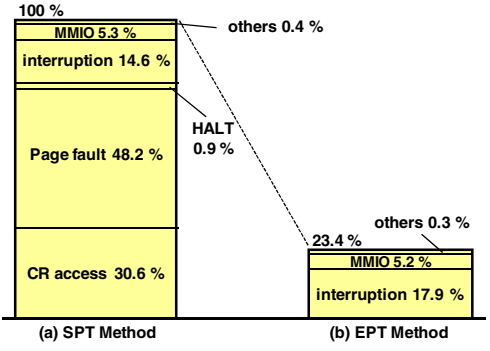


Fig. 5. Performance overhead analysis

It shows that the principal cause of overhead by SPT method is Page Fault and CR access. Page Fault occurs when the address required by running program is not mapped to the physical address. In virtualization environment, a hypervisor translates the address that required by guest program to the physical address on the host side and registers the translation information into SPT when Page Fault occurs. As long as the translation information exists in SPT, when the same page is accessed, processor refers SPT and converts guest address to host physical address. Therefore, Page Fault does not occur. The overhead depends on how often new pages access occur. In this workload, new page access occurred with certain frequency. Consequently, it can be seen a lot of Page Fault overhead. CR access occurs when OS switch the executing process. With the occurrence of CR access, Hypervisor switch SPT. This cause another overhead on CR access if there are many processes switches.

Here we compare it with former research for such analysis of virtualization overhead.

Apparao[13] points out that processing cost increase 600% for context switching when numbers of VM increase from 1 to 4, in their virtualization evaluation environment using Xen hypervisor and SPECjbb2005 workload. It also points out that reasons of the large processing cost are TLB flush, page walk and cache pollution.

Our workload and virtualization software environment is different to above research, but our measurement results takes on similar aspect to it from viewpoint that the primary factor of overhead is management of context switch.

The analysis of here is from a viewpoint of hypervisor software processing, and former research analysis is based on hardware processing factor. That is, the cause of "page fault" and "CR access" in fig. 5 is context switch. Hypervisor makes another SPT for new context, and it changes current page table contents for new SPT, and after that TLB is flushed. Along with execution of new context, page walk process is configured by hardware.

Page fault overhead time measured in fig. 5 includes these particular actions on hypervisor. Similarly CR access on guest OS needs particular actions made by hypervisor. Guest OS accesses logical CR to dispatch new process, hypervisor traps it and emulates for physical CR access with validity check.

EPT method has a feature that processor is able to translate from guest address to host address directly. It is not necessary SPT management. With no hypervisor intervention, no overhead is achieved with EPT method. As a result, if there is a lot of SPT update such as this workload, substantial performance improvement can be expected with applying EPT method.

Fig. 5(b) shows overhead factors of same ERP benchmark workload when the system uses Intel® X5570 processor. Factor of page fault and CR access are suppressed in EPT method, those are appeared in SPT method, and only the overhead which is related to I/O operations like MMIO or interruptions are remain.

At the result, in principle 78.8% of overhead in SPT method is suppressed because of supporting EPT hardware (Fig.5(a)), in practice the overhead of EPT method is smaller 76.6% than SPT method (Fig.5(b)).

This case must be treated carefully as one case, it is not applicable for all cases, but we can understand that EPT method is extremely good control method for achieving low performance overhead.

5 Concluding Remarks

This paper describes a new virtualization technology and presents a comparison between implementations of hardware transparency by software virtualization and hardware-assisted virtualization. In addition, with some performance experiments, we show that Hitachi's virtualization technology, Virtage, has a wide range of practical applications, from the enterprise business area to the high performance computing area.

For achieving its performance, Virtage has performance monitor functions, and it helps resolving much performance problems on virtualization environment.

Hardware transparency on Virtage simplifies complicated system configurations for enterprise systems thanks to the hardware assist features for I/O control. This cannot be achieved with pure software virtualization. The hardware transparency not only enables low emulation overhead from the hypervisor, but it also enables efficient virtualization control, high availability, and high operability.

We examined the performance of virtualization overheads and scalability under different workloads within a Virtage environment. This experiment indicates that Virtage has small virtualization overhead and relatively good performance scalability in practical use with processor over-commitment. Virtage is ready to use for a wide range of applications in the business and high performance computing areas.

However, since with Virtage the guest OS can directly access the hardware specifications of the physical server, it does not meet some user requirements. For example, that a new server installed with a new chipset should be able to support the old version of an OS running on old servers. The new server might be considered unsupported hardware for the old OS.

Our future work will be in the areas of enhancing hardware transparency and satisfying user needs for supporting old OSs on new servers with new chipsets. Also, additional performance evaluation of Virtage in more realistic situations is needed.

References

1. Ueno, H., et al.: Virtage: Hitachi's Virtualization Technology. In: GPC 2009 Workshop Proceedings of Conference, pp. 121–127. IEEE-CS, Los Alamitos (2009)
2. Umeno, H., et al.: Development of a High Performance Virtual Machine System and Performance Measurements for it. IPSJ J. Information Processing 4, 68–78 (1981)
3. Goldberg, R.P.: Survey of Virtual Machine Research. IEEE Computer, 33–45 (1974)
4. Smith, J.E., Nair, R.: The Architecture of Virtual Machines. IEEE Computer, 32–38 (2005)
5. Uhlig, R., et al.: Intel Virtualization Technology. IEEE Computer, 48–56 (2005)
6. Smith, J.E., Nair, R.: Virtual Machines, Versatile Platforms for Systems and Processes. Elsevier Inc., Amsterdam (2005)
7. Barham, A., et al.: Xen and the Art of Virtualization. In: Proc. The 19th ACM Symposium on Operating Systems Principles, pp. 164–177 (2003)
8. Umeno, H., Tanaka, S.: New Methods for Realizing Plural Near-Native Performance Virtual Machines. IEEE Transactions on Computers C-36(9), 1076–1087 (1987)
9. Creasy, R.J.: The Origin of the VM/370 Time-Sharing-System. IBM J. Research and Development, 483–490 (1981)
10. Popek, G.J., Kline, C.S.: The PDP-11 virtual machine architecture: A case study. ACM SIGOPS Operating System Review 9(5), 97–105 (1975)
11. Adair, R., et al.: A Virtual Machine System for the 360/40: Cambridge Scientific Center Report No. G320-2007 (1966)
12. Gil, N., et al.: Intel® Virtualization Technology: Hardware support for efficient processor virtualization. Intel® Virtualization Technology 10(03) (2006)
13. Apparao, P., et al.: Architectural Characterization of VM Scaling on an SMP Machine. In: Min, G., Di Martino, B., Yang, L.T., Guo, M., Rünger, G. (eds.) ISPA Workshops 2006. LNCS, vol. 4331, pp. 464–473. Springer, Heidelberg (2006)